INTERSPEECH 2020
October 25–29, 2020, Shanghai, China

Deep Convolutional Spiking Neural Networks for Keyword Spotting

Emre Yılmaz, Özgür Bora Gevrek, Jibin Wu, Yuxiang Chen, Xuanbo Meng, Haizhou Li

Dept. of Electrical and Computer Engineering, National University of Singapore, Singapore

emrey@kth.se

Abstract

This paper investigates the use of deep convolutional spiking neural networks (SNN) for keyword spotting (KWS) and wakeword detection tasks. The brain-inspired SNN mimic the spike-based information processing of biological neural networks and they can operate on the emerging ultra-low power neuromorphic chips. Unlike conventional artificial neural networks (ANN), SNN process input information asynchronously in an event-driven manner. With temporally sparse input information, this event-driven processing substantially reduces the computational requirements compared to the synchronous computation performed in ANN-based KWS approaches. To explore the effectiveness and computational complexity of SNN on KWS and wakeword detection, we compare the performance and computational costs of spiking fully-connected and convolutional neural networks with ANN counterparts under clean and noisy testing conditions. The results obtained on the Speech Commands and Hey Snips corpora have shown the effectiveness of the convolutional SNN model compared to a conventional CNN with comparable performance on KWS and better performance on the wakeword detection task. With its competitive performance and reduced computational complexity, convolutional SNN models running on energy-efficient neuromorphic hardware offer a low-power and effective solution for mobile KWS applications.

Index Terms: spiking neural networks, keyword spotting, wakeword detection, tandem learning, deep learning

1. Introduction

Accurate and efficient keyword spotting (KWS) is indispensable in developing robust voice interfaces for low-power mobile devices and smart home appliances. KWS systems using deep artificial neural networks (ANN) have made the integration of such voice interfaces into commercial products viable on account of their remarkable performance [1–4]. The improved performance comes with increased computational requirements often due to the time-synchronous processing of input audio signals. Several techniques have been proposed to reduce the computational and memory load of ANN by reducing the number of model parameters [5–8].

The biologically plausible spiking neural networks (SNNs) have attracted great interest in recent years [9]. Their asynchronous and event-driven information processing resembles the computing paradigm observed in human brain, whereby the energy consumption matches to the activity level of the sensory stimuli. Given temporally sparse information transmitted via speech signals, the event-driven computation framework yields tremendous computational efficiency compared to the synchronous computations performed in ANN.

As a non-von Neumann computing paradigm, the neuromorphic computing (NC) implements the event-driven computation of biological neural systems with SNNs in silicon. The emerging NC architectures [10, 11] leverage on the massively parallel, low-power computing units to support spike-based information processing. The design of co-located memory and computing units effectively circumvents the von Neumann bottleneck of low-bandwidth between memory and the processing units [12]. Therefore, integrating the algorithmic power of deep SNNs with the compelling energy efficiency of NC hardware represents an attractive solution for always-on applications such as the wakeword detection.

Due to the discrete and non-differentiable nature of spike generation, the powerful error backpropagation algorithm is not directly applicable to the training of deep spiking neural network. Recently, considerable research efforts are devoted to addressing this problem and the resulting learning rules can be broadly categorized into the SNN-to-ANN conversion [13–16], back-propagation through time with surrogate gradient [17–22], constraint-then-train [23, 24], and tandem learning [25].

Despite several successful attempts on the large-scale image classification tasks with deep SNNs [14–16, 25], speech applications with SNN just to emerge. The first system using a fully-connected two-layer SNN model for KWS on six common words from the TIMIT speech dataset is described in [26]. A detailed benchmarking of keyword spotting efficiency on conventional and neuromorphic devices including a CPU, a GPU, Nvidia's Jetson TX1 accelerator and Intel's Loihi neuromorphic chip is presented in [27]. A recent work of Blouw et al. [28] gives an overview of event-driven signal processing with experimental findings on the Speech Commands dataset.

Recently, we have proposed an SNN-based large-vocabulary automatic speech recognition (ASR) system [29]. Motivated by the promising results on the ASR system, we apply the same tandem learning framework to train SNN-based KWS and wakeword detection systems in this work, and present a comparison of fully-connected and convolutional SNN models with their ANN counterparts in terms of their performance and computational efficiency. To the best of authors' knowledge, this is the first work on the application of deep convolutional SNN models to KWS and wakeword detection tasks.

2. System Description

2.1. Spiking Neuron Model

As shown in Figure 1, the spiking neuron operates asynchronously and integrates the incoming spike trains into its membrane potential. An output spike is generated from the spiking neuron whenever its membrane potential crosses the firing threshold, and this output spike will be propagated to the connected neurons via the axon. The SNNs are constructed by

This research work is supported by Programmatic Grant No. A1687b0033 from the Singapore Government's Research, Innovation and Enterprise 2020 plan (Advanced Manufacturing and Engineering domain). Project Title: Neuromorphic Computing.

http://dx.doi.org/10.21437/Interspeech.2020-1230

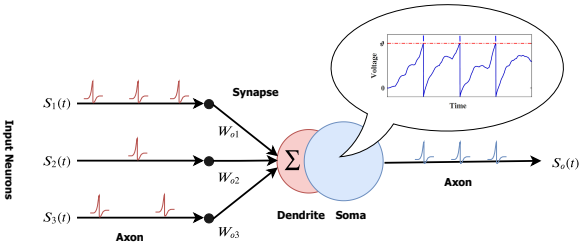


Figure 1: *Illustration of the spiking neuron model*

these spiking neurons that are organized in a feedforward or recurrent structure. Early classification decision can be made from the SNN after the generation of the first output spike. However, the quality of the classification decision is typically improved over time with the accumulating evidence. This is fundamentally different than the synchronous information processing of the conventional ANN, in which the output layer is not activated until all preceding layers are fully updated.

In this work, we use the integrate-and-fire (IF) neuron model with reset by subtraction scheme [14], which can effectively process these quasi-stationary frame-based acoustic features with minimal computational costs. At each time step t of a discrete-time simulation, the incoming spikes to neuron j at layer l are integrated into subthreshold membrane potential V_j^l as follows:

$$V_j^l[t] = V_j^l[t-1] + RI_j^l[t] - \vartheta S_j^l[t-1] \tag{1}$$

with

$$I_j^l[t] = \sum\nolimits_i w_{ji}^{l-1} S_i^{l-1}[t] + b_j^l, \tag{2}$$

where w_{ji}^{l-1} denotes the synaptic weight that connects presynaptic neuron i from layer $l-1$ and b_j^l can be interpreted as a constant injecting current. In addition, $S_i^{l-1}[t]$ indicates the occurrence of an input spike from afferent neuron i at time step t. $I_j^l[t]$ denotes the resulted synaptic current from incoming spike trains. Without loss of generality, a unitary membrane resistance R is assumed here. An output spike is generated whenever $V_j^l[t]$ crosses the firing threshold ϑ as per Eq. 3, which is set to 1 in the experiments by assuming that all synaptic weights are normalized with respect to ϑ. Following a spike generation, the membrane potential is reset by subtracting the firing threshold ϑ as described by the last term of Eq. 1. The $V_j^l[0]$ is reset and initialized to zero for every new input frame,

$$S_j^l[t] = \Theta\left(V_j^l[t] - \vartheta\right) \quad with \quad \Theta(x) = \begin{cases} 1, & if\ x \geq 0 \\ 0, & otherwise. \end{cases} \tag{3}$$

According to Eqs. 1 and 2, the free aggregated membrane potential of neuron j (no firing) in layer l across the simulation time window N_s can be expressed as

$$V_j^{l,f} = \sum\nolimits_i w_{ji}^{l-1} c_i^{l-1} + b_j^l N_s, \tag{4}$$

where c_i^{l-1} is the input spike count from pre-synaptic neuron i at layer $l-1$ as

$$c_i^{l-1} = \sum\nolimits_{t=1}^{N_s} S_i^{l-1}[t]. \tag{5}$$

The $V_j^{l,f}$ summarizes the aggregate membrane potential contributions of the incoming spike trains while ignoring their temporal distribution. As detailed in Section 2.3, this intermediate quantity links the SNN layers to the coupled ANN layers for the parameter optimization.

2.2. Neural Coding Scheme

SNN process information transmitted via spike trains which requires encoding the continuous-valued feature vectors into spike trains at the front-end and perform classification based on the activity of output neurons.

To encode the frame-based input feature vector X (MFCC features in this case) into spike trains, where $X = [x_1, x_2, \cdots, x_n]^T$, we take X as the synaptic current and directly apply it into Eq. 1 at the first time step. Comparing to the commonly used rate coding scheme, whereby the real-valued inputs are sampled into spike trains following a Possion or Bernoulli distribution [15], this neural encoding scheme eliminates the sampling errors. Moreover, it allows the input information to be encoded within the first few time steps that is beneficial for rapid inference. Starting from this neural encoding layer, the spike count c^l and spike train S^l are input to subsequent ANN and SNN layers for tandem learning.

To ensure a smooth learning with high precision error gradients derived at the output layer, instead of using spike count for neural decoding, we use the free aggregate membrane potential of output spiking neurons as the posterior probability for different output classes.

2.3. Training Deep SNNs with Tandem Learning

The tandem learning rule used for training deep SNNs exploits the connection between the activation value of ANN neurons and the spike count of IF neurons. Within the tandem learning framework, a SNN is coupled with an ANN through weight sharing. The SNN layers are used to determine the exact spike representation, which then propagate the aggregate spike counts and spike trains to the subsequent ANN and SNN layers, respectively. It ensures the information that forward propagated to the coupled ANN and SNN layers are synchronized. It is important to note that the ANN is just an auxiliary structure to facilitate the training of SNN and only SNN is used during inference.

As the input features are effectively encoded as spike counts, the temporal structure of the spike trains carries negligible information. Hence, the non-linear transformation of SNN layers can be formulated as

$$c_j^l = f\left(S^{l-1}; w_j^{l-1}, b_j^l\right) \tag{6}$$

where $f()$ denotes the effective transformation performed by spiking neurons. However, an analytical expression from S^{l-1} to c_j^l cannot be determined directly due to the state-dependent nature of spike generation. Therefore, we simplify the spike generation process by assuming the resulting synaptic contributions from S^{l-1} are evenly distributed over the simulation time window. With this assumption, the interspike interval can be determined as follows

$$ISI_j^l = \rho\left(\frac{\vartheta}{V_j^{l,f}/N_s}\right) = \rho\left(\frac{\vartheta}{(\sum\limits_i w_{ji}^{l-1} c_i^{l-1} + b_j^l N_s)/N_s}\right) \tag{7}$$

The approximated 'spike count' a_j^l can be obtained as

$$a_j^l = \frac{N_s}{ISI_j^l} = \frac{1}{\vartheta}\ \rho\left(\sum_i w_{ji}^{l-1} c_i^{l-1} + b_j^l N_s\right) \tag{8}$$

Given a unitary firing threshold ϑ, a_j^l can be effectively determined from an ANN layer of ReLU neurons by setting the spike

count c_i^{l-1} as the input and the aggregated constant injecting current $b_j^l N_s$ as the bias term. This simplification of spike generation process allows the spike-train level error gradients to be approximated from the ANN layer. Algorithmic details of the tandem learning rule can be found in [25].

2.4. SNN-based Keyword Spotting System

With the learning rule described in the previous section, we train a fully-connected and a convolutional SNN model that are designed to detect keywords in fixed-length speech segments. These models are compared with a conventional DNN and CNN model with identical network architectures. The architectures and sizes of both models are chosen in line with the prior work describing CNN-based KWS techniques [30]. The convolutional layers perform 2-D convolution over the acoustic features followed by batch normalization and ReLU activation function. A max-pooling layer is also included in the first convolution layer to reduce dimensionality. All models are applied to a KWS task with multiple keywords and a wakeword detection task with a single target keyword. The former task is a multi-class classification task, while the latter is a detection task with binary output. We further delve into the noise-robustness of these models by performing multi-condition training and explore the performance under clean and noisy scenarios.

3. Experimental Setup

3.1. Datasets

The proposed model is evaluated on the Speech Commands [31] and the 2nd version of the Hey Snips [32] datasets. We use the data preparation setup of the Tensorflow example[1] for reproducibility. This setup prepares 36 922 training, 4445 validation and 4890 test utterances of 1 second duration with 10 target keywords, "yes", "no", "up", "down", "left", "right", "on", "off", "stop" and "go". The remaining 20 commands, "bed", "bird", "cat", "dog", "happy", "house", "Marvin", "Sheila", "tree", "wow" and ten numbers (0-9), are used as unknown words which comprise 10% of the training data. 10% of the training data consists of silence utterances.

The Hey Snips wakeword detection dataset[2] consists of 53 110 training, 8034 development and 7851 test utterances of varying durations spoken by approximately 1.8k speakers. We extract the first second containing speech from each utterance using an energy-based speech activity detection and use these fix-duration utterances in the experiments.

Noisy versions of each dataset are created by artificially adding noise-only segments from the MUSAN corpus [33] at random SNR levels between -5 and 10 dB. The multi-condition training sets are created by combining the clean and noisy training set.

3.2. Implementation details

40-dimensional Mel-frequency cepstral coefficients (MFCC) are extracted with a frame shift of 10ms and a frame length of 30 ms as acoustic features. Resulting 98 frames are stacked to create a 3920-dimensional vector which is fed to the input layer of each neural network model. All neural network models are implemented in PyTorch [34]. The DNN and its spiking counterpart (spikeDNN) has 3 layers, each with 128 hidden units. The CNN and and its spiking counterpart (spikeCNN) has two convolutional layers followed by a single fully connected layer with 100 hidden units. The first convolutional layer has 64 filters with filter size of 20 and 8 in time and frequency dimension, respectively. A 3x3 max-pooling filter has been applied after the first convolutional layer. The second convolutional layer has 32 filters with filter size of 10 and 4 in time and frequency, respectively. Each layer has batch normalization and ReLU activation function. The total number of model parameters is approximately 0.5M for all models. For SNN simulations, all features are encoded within a short time window of 10-time steps.

The neural network training is performed using the Stochastic Gradient Descent (SGD) optimizer with an initial learning rate (LR) of 0.001 and a minibatch size of 100 during clean training and 200 during the multi-condition training. The LR is reduced to 0.0001 after 15 000 training steps and the final model is obtained after 18 000 training steps on the Speech Commands dataset. A similar procedure is followed for training the models on the Hey Snips dataset. Due to faster convergence, the LR is reduced after 4500 training steps and the final model is obtained after 6000 training steps.

3.3. Evaluation Metrics

The keyword detection accuracy is reported using false rejection rate (FRR) vs. false alarm rate (FAR) plots. For the speech commands dataset which contains 10 keywords, we report averaged FRR vs. FAR plots which is obtained by getting the average FRR for all FAR values given by different detection thresholds. To compare the energy efficiency of CNN and spikeCNN, we follow the NC community convention and compute the total synaptic operations SynOps that are required to perform a certain task [11, 14, 15]. For conventional ANN, the total synaptic operations (Multiply-and-Accumulate (MAC)) per classification is defined as SynOps(ANN) $= \sum_{l=1}^{L} f_{in}^l N_l$ where f_{in}^l denotes the number of fan-in connections to each neuron in layer l, and N_l refers to the number of neurons in layer l. L denotes the total number of network layers. For spiking networks, the total synaptic operations (Accumulate (AC)) per classification are correlated with the neurons' firing rate, the number of fan-out connections f_{out} to neurons in the subsequent layer as well as the simulation time window N_s, SynOps(SNN) $= \sum_{t=1}^{N_s} \sum_{l=1}^{L-1} \sum_{j=1}^{N_l} f_{out,j}^l S_j^l[t]$.

4. Results

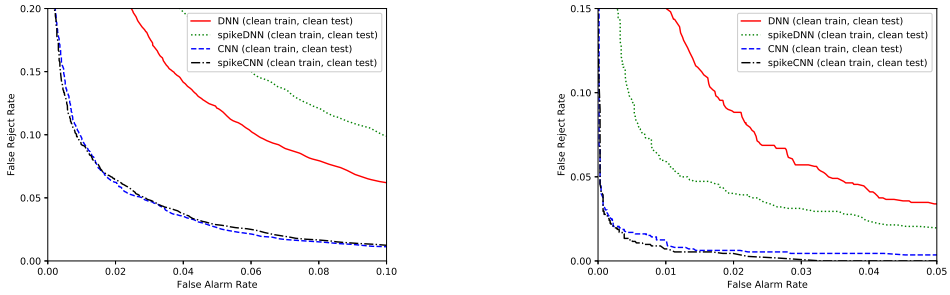
4.1. KWS Results

The KWS results provided by DNN, spikeSNN, CNN and spikeCNN under clean training and testing conditions are given in Figure 2a. The spikeDNN performs significantly worse than the rest and there is a large performance gap with the conventional DNN model. In the presence of convolution layers, the performance gap between the ANN and SNN models diminishes and spikeCNN and CNN provide comparable results under clean conditions.

We further apply multi-condition trained CNN and spikeCNN models to the clean and noisy test set. The performance curves are given in Figure 3a. In general, spikeCNN and CNN models provide comparable results in each condition. Conventional CNN performs marginally better under clean testing scenario, while spikeCNN outperforms the CNN by a narrow margin under noisy testing scenario.

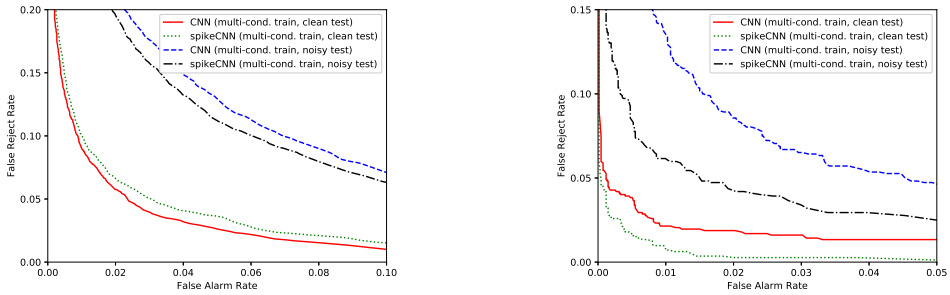
[1] https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/speech_commands

[2] http://research.snips.ai/datasets/keyword-spotting



(a) *Speech Commands KWS task* (b) *Hey Snips wakeword detection task*

Figure 2: *FRR vs. FAR plots given by DNN, spikeDNN, CNN and spikeCNN under clean training and testing conditions*



(a) *Speech Commands KWS task* (b) *Hey Snips wakeword detection task*

Figure 3: *FRR vs. FAR plots given by multi-condition trained CNN and spikeCNN models under clean and noisy testing conditions*

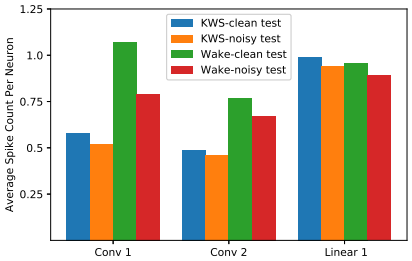


Figure 4: *Average spike count per neuron observed at different spikeCNN layers for different tasks and testing conditions*

4.2. Wakeword Detection Results

When the same models are applied to a wakeword detection task with a binary decision, the spikeDNN model gives superior performance compared to the ANN counterpart. Including convolutional layers reduces both errors substantially, while both spikeCNN and CNN models provide comparable wakeword detection performance. The multi-condition trained CNN and spikeCNN models are first applied to clean test set to quantify the performance degradation compared to the clean trained models. Based on the CNN and spikeCNN results in Figure 2b and 3b, the spikeCNN model is more robust to the mismatch between the training and testing conditions than CNN.

After investigating the clean testing scenario, we move to a more realistic setting with noise-mixed test utterances. Compared to the noisy KWS, the performance gap between spikeCNN and CNN grows indicating the superior performance of convolutional SNN models on the wakeword detection task.

4.3. Computational Efficiency

We report the ratio of average synaptic operations (SynOps(SNN) / SynOps(ANN)) required for the systems compared in Figure 3. The synaptic operations for each system are calculated on a single minibatch of data used during the training. The average spike count per neuron on different task and testing conditions are shown in Figure 4. In general, the neuronal activities are sparse for all network layers given the short encoding time window ($N_s = 10$). On both tasks, we observe slightly reduced neuronal activity at each layer under noisy testing conditions.

For the KWS (wakeword detection) task, the ratio of average synaptic operations is 0.47 and 0.43 (0.90 and 0.69) under clean and noisy testing conditions, respectively. It is important to note that the AC operations performed on SNN are considerably cheaper than the MAC operations required for ANN, e.g. 14 times cheaper and requires 21 times fewer chip area on the Global Foundry 28 nm according to a recent study [14]. Based on this study, the target applications can be performed with 15 to 30 times reduced energy using SNNs on the neuromorphic chips [10, 11].

5. Conclusion

This paper describes a deep convolutional SNN model for KWS and wakeword detection. The spiking models are trained using a recently proposed tandem learning rule which uses coupled ANN-SNN to train SNN models by performing the backpropagation algorithm through the coupled ANN. After the promising results obtained on LVCSR, this work explores the performance and computational efficiency of spiking models trained using the tandem learning rule on KWS and wakeword detection tasks which have various applications for mobile and embedded devices with power restrictions. The experiments performed on the Speech Commands KWS and the Hey Snips wakeword detection dataset demonstrate that the convolutional SNN-based systems provide comparable performance on KWS and better performance on the binary wakeword detection task compared to the conventional CNN models with a ratio of average synaptic operations of 0.45 and 0.8, respectively, corresponding to approximately 15-30 times reduced energy consumption.

6. References

[1] G. Chen, C. Parada, and G. Heigold, "Small-footprint keyword spotting using deep neural networks," in *Proc. ICASSP*, 2014, pp. 4087–4091.

[2] S. Panchapagesan, M. Sun, A. Khare, S. Matsoukas, A. Mandal, B. Hoffmeister, and S. Vitaladevuni, "Multi-task learning and weighted cross-entropy for dnn-based keyword spotting," in *Proc. INTERSPEECH*, 2016, pp. 760–764.

[3] C. Lengerich and A. Hannun, "An end-to-end architecture for keyword spotting and voice activity detection," 2016.

[4] S. Sigtia, R. Haynes, H. Richards, E. Marchi, and J. Bridle, "Efficient voice trigger detection for low resource hardware," in *Proc. INTERSPEECH*, 2018.

[5] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello edge: Keyword spotting on microcontrollers," 2017.

[6] S. Myer and V. S. Tomar, "Efficient keyword spotting using time delay neural networks," in *Proc. INTERSPEECH*, 2018, pp. 1264–1268.

[7] J. Fernández-Marqués, V. W.-S. Tseng, S. Bhattachara, and N. D. Lane, "On-the-fly deterministic binary filters for memory efficient keyword spotting applications on embedded devices," in *Proceedings of the 2nd International Workshop on Embedded and Mobile Deep Learning*, 2018, p. 13–18.

[8] A. Coucke, M. Chlieh, T. Gisselbrecht, D. Leroy, M. Poumeyrol, and T. Lavril, "Efficient keyword spotting using dilated convolutions and gating," in *Proc. ICASSP*, 2019, pp. 6351–6355.

[9] M. Pfeiffer and T. Pfeil, "Deep learning with spiking neurons: Opportunities & challenges," *Frontiers in Neuroscience*, vol. 12, p. 774, 2018.

[10] M. Davies, N. Srinivasa, T. H. Lin, G. Chinya, S. H. Cao, Y.and Choday, G. Dimou, P. Joshi, N. Imam, S. Jain *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, 2018.

[11] P. A. Merolla, J. V. Arthur, R. Alvarez-Icaza, A. S. Cassidy, J. Sawada, F. Akopyan, B. L. Jackson, N. Imam, C. Guo, Y. Nakamura *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014.

[12] D. Monroe, "Neuromorphic computing gets ready for the (really) big time," *Communications of the ACM*, vol. 57, no. 6, pp. 13–15, 2014.

[13] P. U. Diehl, D. Neil, J. Binas, M. Cook, S. C. Liu, and M. Pfeiffer, "Fast-classifying, high-accuracy spiking deep networks through weight and threshold balancing," in *2015 International Joint Conference on Neural Networks (IJCNN)*, July 2015, pp. 1–8.

[14] B. Rueckauer, I. A. Lungu, Y. Hu, M. Pfeiffer, and S. C. Liu, "Conversion of continuous-valued deep networks to efficient event-driven networks for image classification," *Frontiers in Neuroscience*, vol. 11, p. 682, 2017.

[15] A. Sengupta, Y. Ye, R. Wang, C. Liu, and K. Roy, "Going deeper in spiking neural networks: VGG and residual architectures," *Frontiers in Neuroscience*, vol. 13, p. 95, 2019.

[16] Y. Hu, H. Tang, Y. Wang, and G. Pan, "Spiking deep residual network," *arXiv preprint arXiv:1805.01352*, 2018.

[17] E. O. Neftci, H. Mostafa, and F. Zenke, "Surrogate gradient learning in spiking neural networks," *arXiv preprint arXiv:1901.09948*, 2019.

[18] Y. Wu, L. Deng, G. Li, J. Zhu, Y. Xie, and L. Shi, "Direct training for spiking neural networks: Faster, larger, better," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 1311–1318.

[19] S. B. Shrestha and G. Orchard, "Slayer: Spike layer error reassignment in time," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., 2018, pp. 1412–1421.

[20] Y. Wu, L. Deng, G. Li, J. Zhu, and L. Shi, "Spatio-temporal backpropagation for training high-performance spiking neural networks," *Frontiers in Neuroscience*, vol. 12, p. 331, 2018.

[21] G. Bellec, D. Salaj, A. Subramoney, R. Legenstein, and W. Maass, "Long short-term memory and learning-to-learn in networks of spiking neurons," in *Advances in Neural Information Processing Systems*, 2018, pp. 787–797.

[22] F. Zenke and S. Ganguli, "Superspike: Supervised learning in multilayer spiking neural networks," *Neural computation*, vol. 30, no. 6, pp. 1514–1541, 2018.

[23] E. Hunsberger and C. Eliasmith, "Training spiking deep networks for neuromorphic hardware," *arXiv preprint arXiv:1611.05141*, 2016.

[24] J. Wu, Y. Chua, M. Zhang, Q. Yang, G. Li, and H. Li, "Deep spiking neural network with spike count based learning rule," in *2019 International Joint Conference on Neural Networks (IJCNN)*, 2019, pp. 1–6.

[25] J. Wu, Y. Chua, M. Zhang, G. Li, H. Li, and K. C. Tan, "A Tandem Learning Rule for Efficient and Rapid Inference on Deep Spiking Neural Networks," *arXiv e-prints*, p. arXiv:1907.01167, Jul 2019.

[26] B. U. Pedroni, S. Sheik, H. Mostafa, S. Paul, C. Augustine, and G. Cauwenberghs, "Small-footprint spiking neural networks for power-efficient keyword spotting," in *2018 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, 2018, pp. 1–4.

[27] P. Blouw, X. Choo, E. Hunsberger, and C. Eliasmith, "Benchmarking keyword spotting efficiency on neuromorphic hardware," in *Proceedings of the 7th Annual Neuro-Inspired Computational Elements Workshop*, ser. NICE '19, 2019.

[28] P. Blouw and C. Eliasmith, "Event-driven signal processing with neuromorphic computing systems," in *Proc. ICASSP*, 2020, pp. 8534–8538.

[29] J. Wu, E. Yılmaz, M. Zhang, H. Li, and K. C. Tan, "Deep spiking neural networks for large vocabulary automatic speech recognition," *Frontiers in Neuroscience*, vol. 14, p. 199, 2020.

[30] T. N. Sainath and C. Parada, "Convolutional neural networks for small-footprint keyword spotting," in *Proc. INTERSPEECH*, 2015, pp. 1478–1482.

[31] P. Warden, "Speech commands: A dataset for limited-vocabulary speech recognition," 2018.

[32] D. Leroy, A. Coucke, T. Lavril, T. Gisselbrecht, and J. Dureau, "Federated learning for keyword spotting," in *Proc. ICASSP*, 2019, pp. 6341–6345.

[33] D. Snyder, G. Chen, and D. Povey, "MUSAN: A Music, Speech, and Noise Corpus," 2015, arXiv:1510.08484v1.

[34] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, 2019, pp. 8024–8035.